APPENDIX A

```
public class CommonIntSearcher {

  private int[] list1;  // first list of integers in ascending order
  private int[] list2;  // second list of integers in ascending order
  protected int num;    // ?


/**
 *  Creates a new instance of CommonIntSearcher
 */
  public CommonIntSearcher() {}


/**
 *  Finds the index of the entry in _list which is the highest
 *  value just less than _value;
 */
  protected int getIndexJustBelow(int[] _list, int _value) {

      // t => top index
      // b => bottom index
      // m => mid point (index) between 't' and 'b'
      int t = _list.length - 1, b = 0, m = (t+b)/2;

      // loop stops when 't' and 'b' are same or adjacent indexes.
      while(t-b > 1)
      {     if(_list[m] > _value)
            {     t = m;
            }
            else
            {     b = m;
            }
            m = (t+b)/2;
      }
      return b;
  }


/**
 *  Finds the index of the entry in _list which is the highest
 *  value just greater than _value;
 */
  protected int getIndexJustAbove(int[] _list, int _value) {

      // t => top index
      // b => bottom index
      // m => mid point (index) between 't' and 'b'
      int t = _list.length - 1, b = 0, m = (t+b)/2;

      // loop stops when 't' and 'b' are same or adjacent indexes.
      while(t-b > 1)
      {     if(_list[m] > _value)
            {     t = m;
            }
            else
            {     b = m;
            }
            m = (t+b) / 2;
      }
      return t;
  }
```

```
/**
 *  Returns the index of the common integer between list1 and list2.
 */
  public int getIndex(  int _bottom1, int _top1,
                        int _bottom2, int _top2) {

    num++;

    int b1    = _bottom1;      // 'b1' is the bottom index of list 1
    int b2    = _bottom2;      // 'b2' is the bottom index of list 2
    int t1    = _top1;         // 't1' is the top index of list 1
    int t2    = _top2;         // 't1' is the top index of list 2
    int m1, m2;                // 'm1' is the mid point of 'b1', 't1'
                               // 'm2' is the mid point of 'b2', 't2'

    int min1  = list1[b1];     // min1 is value in list 1 at index b1
    int min2  = list2[b2];     // min2 is value in list 2 at index b2
    int max1  = list1[t1];     // max1 is value in list 1 at index t1
    int max2  = list2[t2];     // max2 is value in list 2 at index t2


    m1 = (t1 + b1) /2;         // compute mid point between b1 and t1
    m2 = (t2 + b2) /2;         // compute mid point between b2 and t2


    // check for a common integer (see figure above)
    if(min1 == min2)                 return b1;
    if(max1 == max2)                 return t1;
    if(min1 == max2)                 return b1;
    if(min2 == max1)                 return t1;
    if(list1[m1] == list2[m2])       return m1;

    //  if each list length ≤ 3 & no match => no common integer so
    //  finish after checking corner cases
    if(t1-b1 <= 2 && t2-b2 <= 2)
      {
            if(min1 == list2[m2])    return b1;
            if(min2 == list1[m1])    return m1;
            if(list1[m1] == max2)    return m1;
            if(list2[m2] == max1)    return t1;
            return -1;
      }


      // else bisect the range and look for common int in sub-ranges
      // this requires the top and bottom indices to be recalculated
      if(min1 > min2)
      {     b2 = getIndexJustBelow(list2, min1);
      }
      else
      {     b1 = getIndexJustBelow(list1, min2);
      }

      if(max1 > max2)
      {     t1 = getIndexJustAbove(list1, max2);
      }
      else
      {     t2 = getIndexJustAbove(list2, max1);
      }
```

```
    // compute the new mid indexes of the two sub ranges
    m1 = (b1+t1) / 2;
    m2 = getIndexJustBelow(list2, list1[m1]);


    //  Now being looking for the common integer in the sub ranges.

    // look for the common integer in the first of the new sub ranges
    int index = getIndex(b1, m1-1, b2, m2);

    // if index < 0, then the common int could not have been found in
    // the previous range, so try second range.
    if(index < 0)
      index = getIndex(m1, t1, m2, t2);

    // return the integer of the location of the common integer
    return index;
  }


/**
 *   Returns the common value between the two array lists if one
 *   exists or it returns Integer.MAX_VALUE
 */
  public int getCommonValue(int[] _list1, int[] _list2) {

    list1   = _list1;
    list2   = _list2;
    num     = 0;

    int index = getIndex(0, list1.length-1, 0, list2.length-1);

    if(index < 0) {
      return Integer.MAX_VALUE;
    }
    else {    // the index is from list 1
      return list1[index];
    }
  }

}
```